



Introduction to *Code_Saturne*

J. Uribe

¹EDF Energy, R&D UK Centre





- 1 Introduction
- 2 Help and info
- 3 Preprocessing
- 4 Running a case
- 5 Postprocessing





Code_Saturne is a 3D unstructured finite volume code developed by Electricité de France (EDF) since 2000. It was made open-source in 2007 under the GPL license.

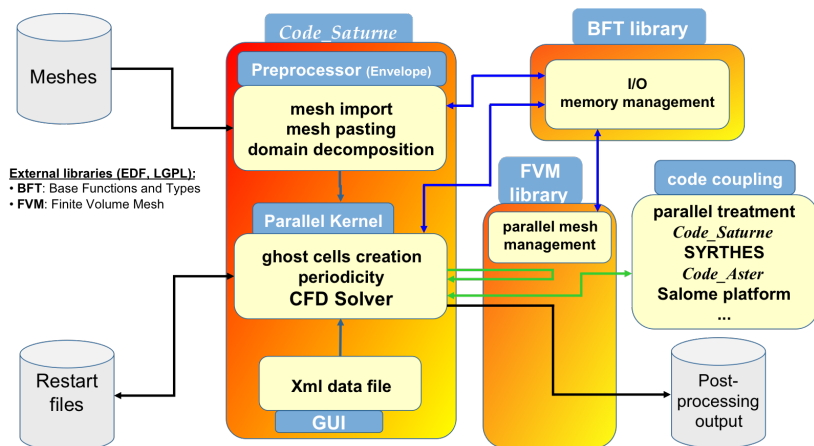
- Single phase turbulent flows:
 $k - \varepsilon$, $k - \omega$, $\overline{v^2} - f$, $R_{ij} - \varepsilon$, *LES*.
- Radiative heat transfer.
- Combustion coal, heavy fuel oil, gas.
- Electric arc and Joule effect.
- Lagrangian module for dispersed particle tracking.
- Compressible flow.
- ALE method for deformable meshes.
- Conjugate heat transfer (SYRTHES and 1D module).
- Specific modules for nuclear waste storage and cooling towers.



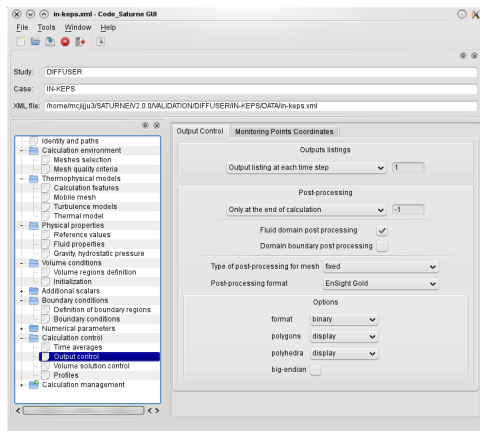
- Written in FORTRAN (49%), C (41%) and Python (10%).
- Graphical User Interface available in Qt (xml format files).
- Parallel coding using mpi.
- Each release version is validated in around 30 cases with 1 to 15 simulations per case.
- Highly portable: Linux/UNIX, Solaris, Irix64, HP-UX, Mac OS.
- Validated on different architectures: Opteron, Itanium, Power5-6, BlueGene.



Structure of *Code_Saturne*



- Graphical interface written in Qt.
- Allows to set up parameters for the case.
- Generates a xml file.
- v3.0 has Mathematical Expression Interpreter.
- Not all specific physics available.
- User subroutines have precedence.





In version 3.0 the command `code_saturne` accesses all scripts of the code:

```
[mcjijju3@flair4gba:bin$] code_saturne
```

```
Usage: code_saturne <topic>
```

Topics:

- help
- autovnv
- compile
- config
- create
- gui
- info
- run
- salome

Options:

- h, --help show this help message and exit



In version 3.0 the command `code_saturne` accesses all scripts of the code:

```
[mcjijju3@flair4gba:bin$] code_saturne info -help
Usage: cs info [options]
Options:
  -h, --help                show this help message
                           and exit
  -r <pdfreader>, --reader=<pdfreader>
                           define a pdf reader
  -g <guide>, --guide=<guide>
                           open a manual ['install', 'refcard', 'tutorial',
                           'theory', 'developer', 'autovnv', 'user']
  --version                  print version number
```



In version 3.0 the command `code_saturne` accesses all scripts of the code:

```
[mcjijju3@flair4gba:bin$] code_saturne info -help
Usage: cs info [options]
Options:
  -h, --help                show this help message
                           and exit
  -r <pdfreader>, --reader=<pdfreader>
                           define a pdf reader
  -g <guide>, --guide=<guide>
                           open a manual ['install', 'refcard', 'tutorial',
                           'theory', 'developer', 'autovnv', 'user']
  --version                 print version number
```

The command `code_saturne info -g user` will bring the pdf version of the user's manual.





In version 3.0 the command `code_saturne` accesses all scripts of the code:

```
[mcjijju3@flair4gba:bin$] code_saturne info -help
Usage: cs info [options]
Options:
  -h, --help                show this help message
                           and exit
  -r <pdfreader>, --reader=<pdfreader>
                           define a pdf reader
  -g <guide>, --guide=<guide>
                           open a manual ['install', 'refcard', 'tutorial',
                           'theory', 'developer', 'autovnv', 'user']
  --version                 print version number
```

The command `code_saturne info -g user` will bring the pdf version of the user's manual.

The manual can be access from **Help** menu of the graphical interface.



In version 3.0 the command `code_saturne` accesses all scripts of the code:

```
[mcjijju3@flair4gba:bin$] code_saturne info -help
Usage: cs info [options]
Options:
  -h, --help                show this help message
                           and exit
  -r <pdfreader>, --reader=<pdfreader>
                           define a pdf reader
  -g <guide>, --guide=<guide>
                           open a manual ['install', 'refcard', 'tutorial',
                           'theory', 'developer', 'autovnv', 'user']
  --version                 print version number
```

The command `code_saturne info -g user` will bring the pdf version of the user's manual.

The manual can be access from **Help** menu of the graphical interface.

More help via support email saturne-support@edf.fr or on the forums: <http://code-saturne.org>

No mesh generator included in *Code_Saturne* but it can read a wide range of formats:

- Universal I-DEAS (.unv).
From NX or ICEM software.
- Hexa NUMECA (.hex)
- MED 2.2 (.med). From Salomé platform
- CGNS (.cgns). Generic format.
- Gmesh.
- Gambit Neutral (.neu).
- StarCD (.ccm). Via libccmio available from StarCD on request.

NX **SIEMENS**

SALOME





The *Code_Saturne* preprocessor works independently from the Kernel. To check your mesh click on the GUI button





The *Code_Saturne* preprocessor works independently from the Kernel. To check your mesh click on the GUI button

- Checks all aspects of the mesh, including periodicity.





The *Code_Saturne* preprocessor works independently from the Kernel. To check your mesh click on the GUI button

- Checks all aspects of the mesh, including periodicity.
- Lists all boundaries.





The *Code_Saturne* preprocessor works independently from the Kernel. To check your mesh click on the GUI button

- Checks all aspects of the mesh, including periodicity.
- Lists all boundaries.
- Quality information.





The *Code_Saturne* preprocessor works independently from the Kernel. To check your mesh click on the GUI button

- Checks all aspects of the mesh, including periodicity.
- Lists all boundaries.
- Quality information.
- Creates Enight files for visualisation that can be open with ParaView.





Code_Saturne needs specific directory structure. It is organised in **STUDY/CASE** hierarchy.





Code_Saturne needs specific directory structure. It is organised in **STUDY/CASE** hierarchy.

- To create the necessary structure use the command:
`code_saturne create -s STUDY1 -c CASE1 -c CASE2`
... which creates a study named **STUDY1** with cases **CASE1** and **CASE2**.





Code_Saturne needs specific directory structure. It is organised in **STUDY/CASE** hierarchy.

- To create the necessary structure use the command:
`code_saturne create -s STUDY1 -c CASE1 -c CASE2`
... which creates a study named **STUDY1** with cases **CASE1** and **CASE2**.
- Cases can be added by using:
`code_saturne create -c CASE3`





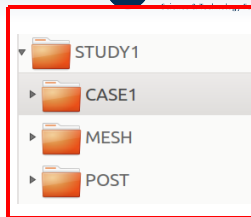
Code_Saturne needs specific directory structure. It is organised in **STUDY/CASE** hierarchy.

- To create the necessary structure use the command:
`code_saturne create -s STUDY1 -c CASE1 -c CASE2`
... which creates a study named **STUDY1** with cases **CASE1** and **CASE2**.
- Cases can be added by using:
`code_saturne create -c CASE3`
- More info by typing:
`code_saturne create -h`



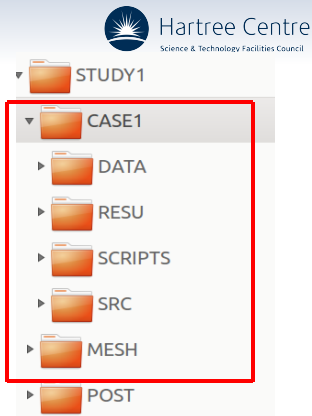
Running a case - File structure

- **STUDY** contains the cases, the mesh and a post-processing (empty) directories.



Running a case - File structure

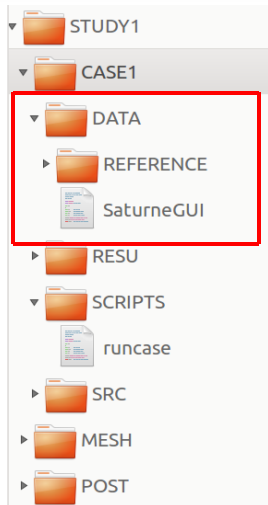
- STUDY contains the cases, the mesh and a post-processing (empty) directories.
- CASE** contains all input data, running scripts and results files.



Running a case - File structure



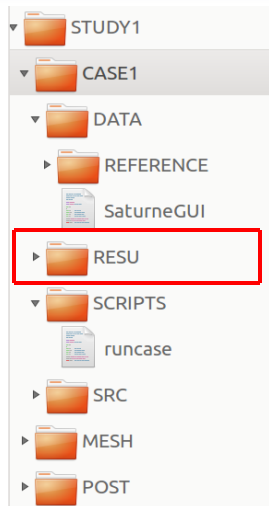
- STUDY contains the cases, the mesh and a post-processing (empty) directories.
- CASE contains all input data, running scripts and results files.
- **DATA** has the graphical interface script and xml file.



Running a case - File structure



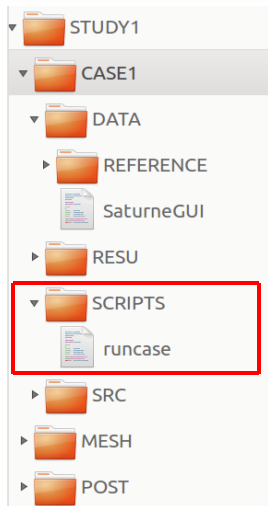
- STUDY contains the cases, the mesh and a post-processing (empty) directories.
- CASE contains all input data, running scripts and results files.
- DATA has the graphical interface script and xml file.
- **RESU** has the saved results after the simulation has finished.



Running a case - File structure



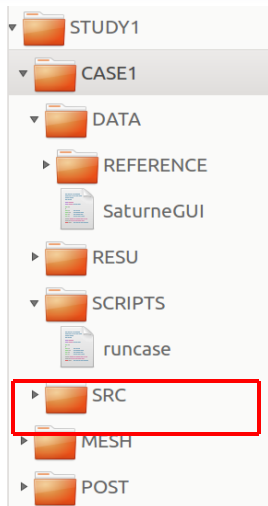
- STUDY contains the cases, the mesh and a post-processing (empty) directories.
- CASE contains all input data, running scripts and results files.
- DATA has the graphical interface script and xml file.
- RESU has the saved results after the simulation has finished.
- **SCRIPT** contains the runcase file to launch the calculation.



Running a case - File structure



- STUDY contains the cases, the mesh and a post-processing (empty) directories.
- CASE contains all input data, running scripts and results files.
- DATA has the graphical interface script and xml file.
- RESU has the saved results after the simulation has finished.
- SCRIPT contains the runcase file to launch the calculation.
- **SRC** contains the user subroutines.



Running a case - GUI



Hartree Centre
Science & Technology Facilities Council

To use the graphical interface go to the **DATA** directory and execute **./SaturneGUI**

- Create a new file.

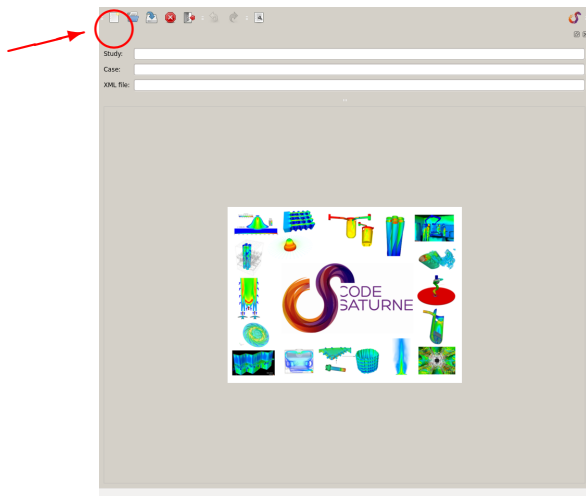


Running a case - GUI



Hartree Centre
Science & Technology Facilities Council

To use the graphical interface go to the **DATA** directory and execute **./SaturneGUI**



Running a case - GUI



Hartree Centre
Science & Technology Facilities Council

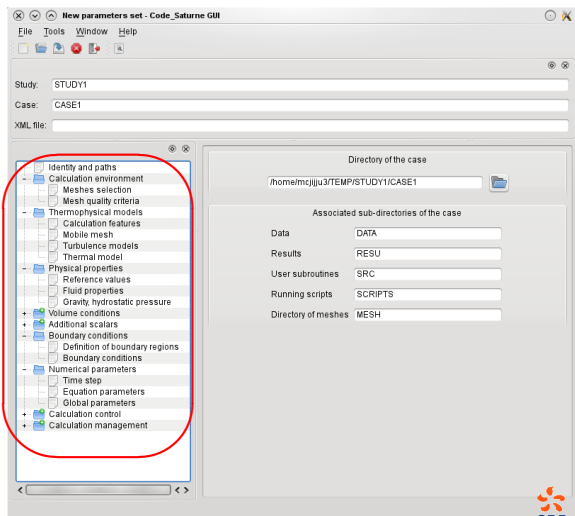
To use the graphical interface go to the **DATA** directory and execute **./SaturneGUI**

- Create a new file.
- Go through the folders and fill the necessary information for the case (physical properties, boundary conditions, numerical options, etc.).



Running a case - GUI

To use the graphical interface go to the **DATA** directory and execute **./SaturneGUI**



Running a case - GUI



Hartree Centre
Science & Technology Facilities Council

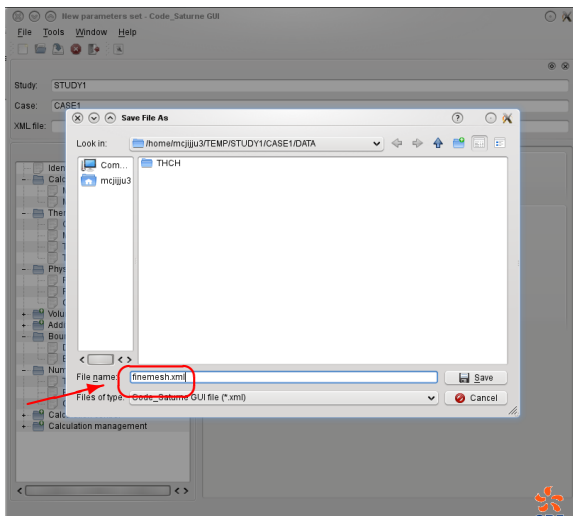
To use the graphical interface go to the **DATA** directory and execute **./SaturneGUI**

- Create a new file.
- Go through the folders and fill the necessary information for the case (physical properties, boundary conditions, numerical options, etc.).
- Save an .xml file (by default in the data directory).



Running a case - GUI

To use the graphical interface go to the **DATA** directory and execute **./SaturneGUI**



Running a case - GUI



Hartree Centre
Science & Technology Facilities Council

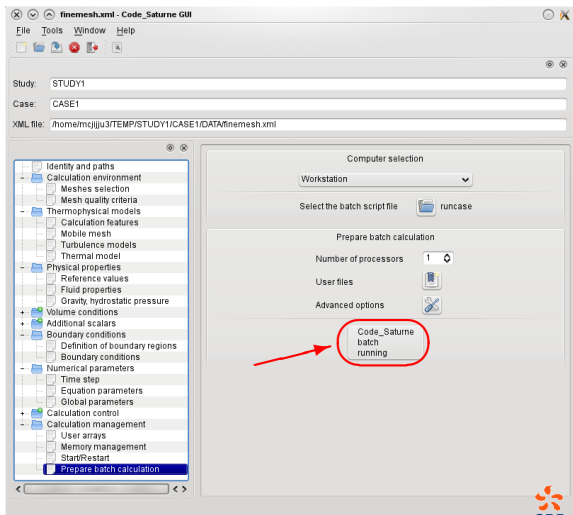
To use the graphical interface go to the **DATA** directory and execute **./SaturneGUI**

- Create a new file.
- Go through the folders and fill the necessary information for the case (physical properties, boundary conditions, numerical options, etc.).
- Save an .xml file (by default in the data directory).
- Click on **Code_Saturne batch running** and check the terminal.



Running a case - GUI

To use the graphical interface go to the **DATA** directory and execute **./SaturneGUI**

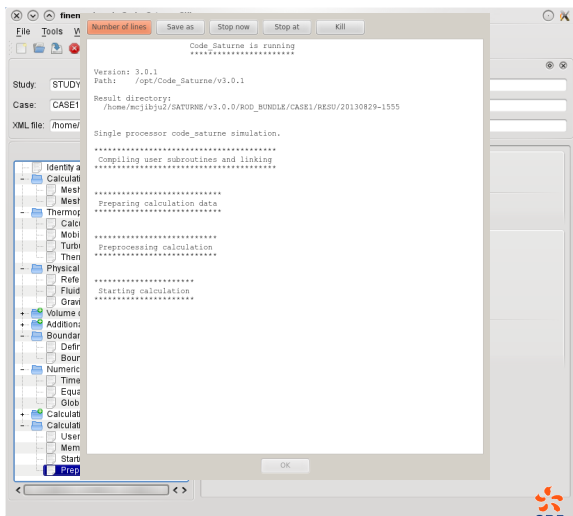


Running a case - GUI



Hartree Centre
Science & Technology Facilities Council

To use the graphical interface go to the **DATA** directory and execute **./SaturneGUI**



Running a case - GUI



Hartree Centre
Science & Technology Facilities Council

To use the graphical interface go to the **DATA** directory and execute **./SaturneGUI**

- Create a new file.
- Go through the folders and fill the necessary information for the case (physical properties, boundary conditions, numerical options, etc.).
- Save an .xml file (by default in the data directory).
- Click on **Code_Saturne batch running** and check the terminal.
- Not all modules are available on the GUI. Sometimes use of the FORTRAN files is required as complement.





If the GUI is not used, all necessary parameters have to be filled in the FORTRAN subroutines.

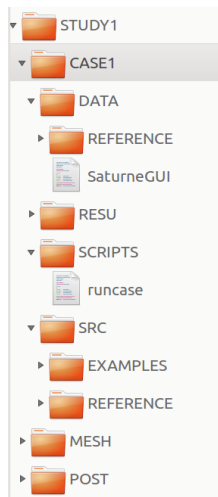


Running a case - FORTRAN



If the GUI is not used, all necessary parameters have to be filled in the FORTRAN subroutines.

- All used subroutines have to be copied into the **SRC** directory from the **REFERENCE** sub-directory.

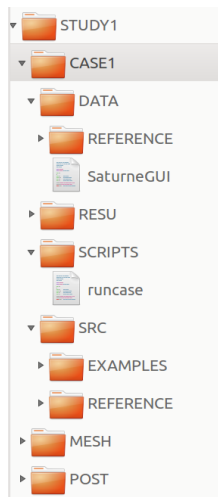


Running a case - FORTRAN



If the GUI is not used, all necessary parameters have to be filled in the FORTRAN subroutines.

- All used subroutines have to be copied into the **SRC** directory from the **REFERENCE** sub-directory.
- There are good examples in the **EXAMPLES** sub-directory.

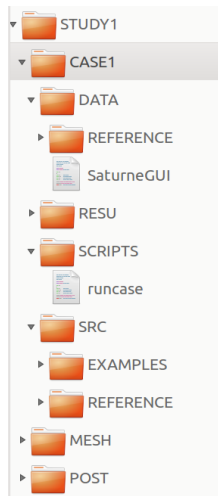


Running a case - FORTRAN



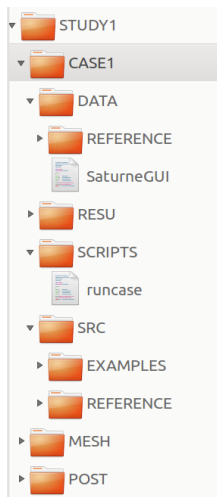
If the GUI is not used, all necessary parameters have to be filled in the FORTRAN subroutines.

- All used subroutines have to be copied into the **SRC** directory from the **REFERENCE** sub-directory.
- There are good examples in the **EXAMPLES** sub-directory.



If the GUI is not used, all necessary parameters have to be filled in the FORTRAN subroutines.

- All used subroutines have to be copied into the **SRC** directory from the **REFERENCE** sub-directory.
- There are good examples in the **EXAMPLES** sub-directory.
- At least **cs_user_parameters.f90** (main physical and numerical parameters) and **cs_user_boundary_conditions.f90** (boundary conditions) are mandatory if not using the GUI.

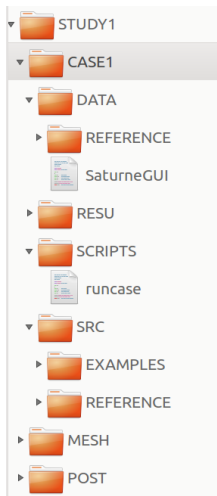


Running a case - FORTRAN



If the GUI is not used, all necessary parameters have to be filled in the FORTRAN subroutines.

- All used subroutines have to be copied into the **SRC** directory from the **REFERENCE** sub-directory.
- There are good examples in the **EXAMPLES** sub-directory.
- At least **cs_user_parameters.f90** (main physical and numerical parameters) and **cs_user_boundary_conditions.f90** (boundary conditions) are mandatory if not using the GUI.
- FORTRAN files have precedence over the GUI xml file.



Running a case - FORTRAN



Once all subroutines are complete, go to the **DATA** directory and copy **cs_user_scripts.py** from the **REFERENCE** directory and edit it.



```
#!/usr/bin/env python

#-----
# This file is part of Code_Saturne, a general-purpose CFD tool.
#
# Copyright (C) 1998-2013 EDF S.A.
#
# This program is free software; you can redistribute it and/or modify it under
# the terms of the GNU General Public License as published by the Free Software
# Foundation; either version 2 of the License, or (at your option) any later
# version.
#
# This program is distributed in the hope that it will be useful, but WITHOUT
# ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS
# FOR A PARTICULAR PURPOSE. See the GNU General Public License for more
# details.
#
# You should have received a copy of the GNU General Public License along with
# this program; if not, write to the Free Software Foundation, Inc., 51 Franklin
# Street, Fifth Floor, Boston, MA 02110-1301, USA.
#-----

import os

#=====
# Local functions
#=====

def domain_auto_restart(domain, n_add):
    """
    Select latest valid checkpoint for restart, and
    create f1cstp file to add n_add time steps.
    """

    from cs_exec_environment import get_command_output

    results_dir = os.path.abspath(os.path.join(self.result_dir, '..'))
    results = os.listdir(results_dir)
    results.sort(reverse=True)
    for r in results:
        m = os.path.join(results_dir, r, 'checkpoint', 'main')
        if os.path.isfile(m):
            try:
                cmd = self.package.get_io_dump()
                cmd += ' --section nbre_pas_de_temps --extract ' + m
```



Once all subroutines are complete, go to the **DATA** directory and copy **cs_user_scripts.py** from the **REFERENCE** directory and edit it.

- Add the name of the mesh file.



```
#!/usr/bin/env python

#-----
# This file is part of Code_Saturne, a general-purpose CFD tool.
#
# Copyright (C) 1998-2013 EDF S.A.
#
# This program is free software; you can redistribute it and/or modify it under
# the terms of the GNU General Public License as published by the Free Software
# Foundation; either version 2 of the License, or (at your option) any later
# version.
#
# This program is distributed in the hope that it will be useful, but WITHOUT
# ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS
# FOR A PARTICULAR PURPOSE. See the GNU General Public License for more
# details.
#
# You should have received a copy of the GNU General Public License along with
# this program; if not, write to the Free Software Foundation, Inc., 51 Franklin
# Street, Fifth Floor, Boston, MA 02110-1301, USA.
#-----

import os

#=====
# Local functions
#=====

def domain_auto_restart(domain, n_add):
    """
    Select latest valid checkpoint for restart, and
    create f1cstp file to add n_add time steps.
    """

    from cs_exec_environment import get_command_output

    results_dir = os.path.abspath(os.path.join(self.result_dir, '..'))
    results = os.listdir(results_dir)
    results.sort(reverse=True)
    for r in results:
        m = os.path.join(results_dir, r, 'checkpoint', 'main')
        if os.path.isfile(m):
            try:
                cmd = self.package.get_io_dump()
                cmd += ' --section nbre_pas_de_temps --extract ' + m
```



Once all subroutines are complete, go to the **DATA** directory and copy **cs_user_scripts.py** from the **REFERENCE** directory and edit it.

- Add the name of the mesh file.
- Add location of restart files if required.



```
#!/usr/bin/env python

#-----
# This file is part of Code_Saturne, a general-purpose CFD tool.
#
# Copyright (C) 1998-2013 EDF S.A.
#
# This program is free software; you can redistribute it and/or modify it under
# the terms of the GNU General Public License as published by the Free Software
# Foundation; either version 2 of the License, or (at your option) any later
# version.
#
# This program is distributed in the hope that it will be useful, but WITHOUT
# ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS
# FOR A PARTICULAR PURPOSE. See the GNU General Public License for more
# details.
#
# You should have received a copy of the GNU General Public License along with
# this program; if not, write to the Free Software Foundation, Inc., 51 Franklin
# Street, Fifth Floor, Boston, MA 02110-1301, USA.
#-----

import os

#-----
# Local functions
#-----

def domain_auto_restart(domain, n_add):
    """
    Select latest valid checkpoint for restart, and
    create f1cstp file to add n_add time steps.
    """

    from cs_exec_environment import get_command_output

    results_dir = os.path.abspath(os.path.join(self.result_dir, '..'))
    results = os.listdir(results_dir)
    results.sort(reverse=True)
    for r in results:
        m = os.path.join(results_dir, r, 'checkpoint', 'main')
        if os.path.isfile(m):
            try:
                cmd = self.package.get_io_dump()
                cmd += ' --section nbre_pas_de_temps --extract ' + m
```

Once all subroutines are complete, go to the **DATA** directory and copy **cs_user_scripts.py** from the **REFERENCE** directory and edit it.

- Add the name of the mesh file.
- Add location of restart files if required.
- Add number of processors if running in parallel.



```
#!/usr/bin/env python

#-----
# This file is part of Code_Saturne, a general-purpose CFD tool.
#
# Copyright (C) 1998-2013 EDF S.A.
#
# This program is free software; you can redistribute it and/or modify it under
# the terms of the GNU General Public License as published by the Free Software
# Foundation; either version 2 of the License, or (at your option) any later
# version.
#
# This program is distributed in the hope that it will be useful, but WITHOUT
# ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS
# FOR A PARTICULAR PURPOSE. See the GNU General Public License for more
# details.
#
# You should have received a copy of the GNU General Public License along with
# this program; if not, write to the Free Software Foundation, Inc., 51 Franklin
# Street, Fifth Floor, Boston, MA 02110-1301, USA.
#-----

import os

#=====
# Local functions
#=====

def domain_auto_restart(domain, n_add):
    """
    Select latest valid checkpoint for restart, and
    create file to add n_add time steps.
    """

    from cs_exec_environment import get_command_output

    results_dir = os.path.abspath(os.path.join(self.result_dir, '..'))
    results = os.listdir(results_dir)
    results.sort(reverse=True)
    for r in results:
        m = os.path.join(results_dir, r, 'checkpoint', 'main')
        if os.path.isfile(m):
            try:
                cmd = self.package.get_io_dump()
                cmd += ' --section nbre_pas_de_temps --extract ' + m
```

Once all subroutines are complete, go to the **DATA** directory and copy **cs_user_scripts.py** from the **REFERENCE** directory and edit it.

- Add the name of the mesh file.
- Add location of restart files if required.
- Add number of processors if running in parallel.
- Other parameters can also be edited if necessary.



```
#!/usr/bin/env python

#-----
# This file is part of Code_Saturne, a general-purpose CFD tool.
#
# Copyright (C) 1998-2013 EDF S.A.
#
# This program is free software; you can redistribute it and/or modify it under
# the terms of the GNU General Public License as published by the Free Software
# Foundation; either version 2 of the License, or (at your option) any later
# version.
#
# This program is distributed in the hope that it will be useful, but WITHOUT
# ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS
# FOR A PARTICULAR PURPOSE. See the GNU General Public License for more
# details.
#
# You should have received a copy of the GNU General Public License along with
# this program; if not, write to the Free Software Foundation, Inc., 51 Franklin
# Street, Fifth Floor, Boston, MA 02110-1301, USA.
#-----

import os

#=====
# Local functions
#=====

def domain_auto_restart(domain, n_add):
    """
    Select latest valid checkpoint for restart, and
    create f1cstp file to add n_add time steps.
    """

    from cs_exec_environment import get_command_output

    results_dir = os.path.abspath(os.path.join(self.result_dir, '..'))
    results = os.listdir(results_dir)
    results.sort(reverse=True)
    for r in results:
        m = os.path.join(results_dir, r, 'checkpoint', 'main')
        if os.path.isfile(m):
            try:
                cmd = self.package.get_io_dump()
                cmd += ' --section nbre_pas_de_temps --extract ' + m
```

Once all subroutines are complete, go to the **DATA** directory and copy **cs_user_scripts.py** from the **REFERENCE** directory and edit it.

- Add the name of the mesh file.
- Add location of restart files if required.
- Add number of processors if running in parallel.
- Other parameters can also be edited if necessary.
- Run the script **./runcase** under **SCRIPTS**.



```
#!/usr/bin/env python

#-----
# This file is part of Code_Saturne, a general-purpose CFD tool.
#
# Copyright (C) 1998-2013 EDF S.A.
#
# This program is free software; you can redistribute it and/or modify it under
# the terms of the GNU General Public License as published by the Free Software
# Foundation; either version 2 of the License, or (at your option) any later
# version.
#
# This program is distributed in the hope that it will be useful, but WITHOUT
# ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS
# FOR A PARTICULAR PURPOSE. See the GNU General Public License for more
# details.
#
# You should have received a copy of the GNU General Public License along with
# this program; if not, write to the Free Software Foundation, Inc., 51 Franklin
# Street, Fifth Floor, Boston, MA 02110-1301, USA.
#-----

import os

#-----
# Local functions
#-----

def domain_auto_restart(domain, n_add):
    """
    Select latest valid checkpoint for restart, and
    create f1cstp file to add n_add time steps.
    """

    from cs_exec_environment import get_command_output

    results_dir = os.path.abspath(os.path.join(self.result_dir, '..'))
    results = os.listdir(results_dir)
    results.sort(reverse=True)
    for r in results:
        m = os.path.join(results_dir, r, 'checkpoint', 'main')
        if os.path.isfile(m):
            try:
                cmd = self.package.get_io_dump()
                cmd += ' --section nbre_pas_de_temps --extract ' + m
```



- After running the **runcase** either from the terminal or through the GUI, check the output window.

```
[mcjijju3@flair4gba:SCRIPTS$] ./runcase
      Code_Saturne is running
      *****

Version: 3.0.1
Path:    /opt/Code_Saturne/v3.0.1
Result directory:
    /home/mcjibju2/SATURNE/v3.0.0/STUDY1/CASE1/RESU/20130829-1725
Single processor code_saturne simulation.
*****
    Compiling user subroutines and linking
*****
*****
    Preparing calculation data
*****
*****
    Preprocessing calculation
*****
```



- After running the **runcase** either from the terminal or through the GUI, check the output window.
- The calculation will run on a directory with the date under **RESU**.

```
[mcjijju3@flair4gba:SCRIPTS$] ./runcase
Code_Saturne is running
*****
Version: 3.0.1
Path: /opt/Code_Saturne/v3.0.1
Result directory:
/home/mcjibju2/SATURNE/v3.0.0/STUDY1/CASE1/RESU/20130829-1725
Single processor code_saturne simulation.
*****
Compiling user subroutines and linking
*****
*****
Preparing calculation data
*****
*****
Preprocessing calculation
*****
```



- After running the **run**case either from the terminal or through the GUI, check the output window.
- The calculation will run on a directory with the date under **RESU**.
- The listing file contains all input parameters and information on the variables at each iteration.

```
[mcjijju3@flair4gba:SCRIPTS$] ./runcase
                        Code_Saturne is running
                        *****

Version: 3.0.1
Path:      /opt/Code_Saturne/v3.0.1
Result directory:
           /home/mcjibju2/SATURNE/v3.0.0/STUDY1/CASE1/RESU/20130829-1725
Single processor code_saturne simulation.
*****
Compiling user subroutines and linking
*****
*****
Preparing calculation data
*****
*****
Preprocessing calculation
*****
```



- After running the **runcase** either from the terminal or through the GUI, check the output window.
- The calculation will run on a directory with the date under **RESU**.
- The `listing` file contains all input parameters and information on the variables at each iteration.
- Once the calculation is finished, all relevant files are copied to the **RESU** directory.

```
[mcjijju3@flair4gba:SCRIPTS] ls ../RESU/20130829-1734/  
checkpoint compile.log listing mesh_output  
monitoring performance.log postprocessing  
preprocessor.log rod_budle.xml setup.log src_saturne summary
```

In case of problems ...



Hartree Centre
Science & Technology Facilities Council





- Check compilation log.

```
[mcjijju3@flair4gba:SCRIPTS$] ./runcase
*****
Compiling user subroutines and linking
*****

Traceback (most recent call last):
  File "/opt/Code_Saturne/v3.0.1/bin/code_saturne", line 77, in <module>
    retcode = cs.execute()
  File
    .
    .
    .
"/opt/Code_Saturne/v3.0.1/lib/python2.7/site-packages/code_saturne/cs_case_domain.py", line 578, in compile_and_link
    raise RunCaseError('Compile or link error.')
cs_case_domain.RunCaseError: Compile or link error.

[mcjijju3@flair4gba:SCRIPTS$] ls ../RESU/20130829-1735/
compil.log src_saturne
[mcjijju3@flair4gba:SCRIPTS$]
```



- Check compilation log.
- Check preprocessing output file `preprocessor.log`.

```
[mcjijju3@flair4gba:SCRIPTS$] ./runcase
                        Code_Saturne is running
                        *****

Version: 3.0.1
Path:      /opt/Code_Saturne/v3.0.1
Result directory:
  /home/mcjibju2/STUDY1//CASE1/RESU/20130829-1748
Single processor code_saturne simulation.
*****
  Preparing calculation data
*****
*****
  Preprocessing calculation
*****
Error running the preprocessor.
Check the preprocessor.log file for details.
Error in preprocessing stage.
*****
  Saving calculation results
*****
Error in preprocessing stage.
```



- Check compilation log.
- Check preprocessing output file `preprocessor.log`.
- Check listing file.

```
[mcjijju3@flair4gba:SCRIPTS$] ./runcase
Code_Saturne is running
*****
Version: 3.0.1
Path: /opt/Code_Saturne/v3.0.1
Result directory:
/home/mcjibju2/STUDY1/CASE1/RESU/20130829-1751
Single processor code_saturne simulation.
*****
Preparing calculation data
*****
*****
Preprocessing calculation
*****
*****
Starting calculation
*****
solver script exited with status 1.
Error running the calculation.
Check code_saturne log (listing) and error* files for details.
```



After the calculation has finished, The **RESU** directory contains directories by date: YYYYMMDD-HHMM (i.e. 20130829-1748):

- **postprocessing**: Has the files for visualisation using Ensignt or ParaView.





After the calculation has finished, The **RESU** directory contains directories by date: YYYYMMDD-HHMM (i.e. 20130829-1748):

- postprocessing: Has the files for visualisation using Ensign or ParaView.
- **monitoring**: Contains history files of variables at given locations (probes).





After the calculation has finished, The **RESU** directory contains directories by date: YYYYMMDD-HHMM (i.e. 20130829-1748):

- postprocessing: Has the files for visualisation using Ensight or ParaView.
- monitoring: Contains history files of variables at given locations (probes).
- **checkpoint**: Has the necessary files to restart a calculation.





After the calculation has finished, The **RESU** directory contains directories by date: YYYYMMDD-HHMM (i.e. 20130829-1748):

- postprocessing: Has the files for visualisation using Ensight or ParaView.
- monitoring: Contains history files of variables at given locations (probes).
- checkpoint: Has the necessary files to restart a calculation.
- **src_saturne**: A copy of the FORTRAN files used to run the





After the calculation has finished, The **RESU** directory contains directories by date: YYYYMMDD-HHMM (i.e. 20130829-1748):

- postprocessing: Has the files for visualisation using Ensight or ParaView.
- monitoring: Contains history files of variables at given locations (probes).
- checkpoint: Has the necessary files to restart a calculation.
- src_saturne: A copy of the FORTRAN files used to run the
- Any user output files.





After the calculation has finished, The **RESU** directory contains directories by date: YYYYMMDD-HHMM (i.e. 20130829-1748):

- postprocessing: Has the files for visualisation using Ensight or ParaView.
- monitoring: Contains history files of variables at given locations (probes).
- checkpoint: Has the necessary files to restart a calculation.
- src_saturne: A copy of the FORTRAN files used to run the
- Any user output files.
- preprocessor.log and listing files for the preprocessor and kernel executions.





After the calculation has finished, The **RESU** directory contains directories by date: YYYYMMDD-HHMM (i.e. 20130829-1748):

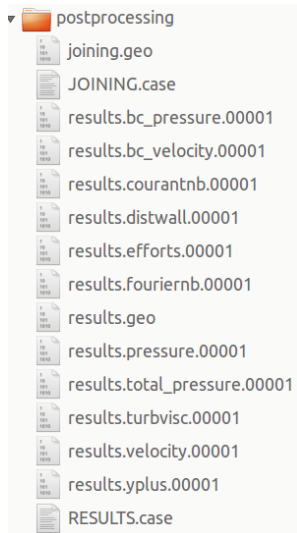
- postprocessing: Has the files for visualisation using Ensight or ParaView.
- monitoring: Contains history files of variables at given locations (probes).
- checkpoint: Has the necessary files to restart a calculation.
- src_saturne: A copy of the FORTRAN files used to run the
- Any user output files.
- preprocessor.log and listing files for the preprocessor and kernel executions.
- Copy of the .xml file used.



Results: postprocessing files



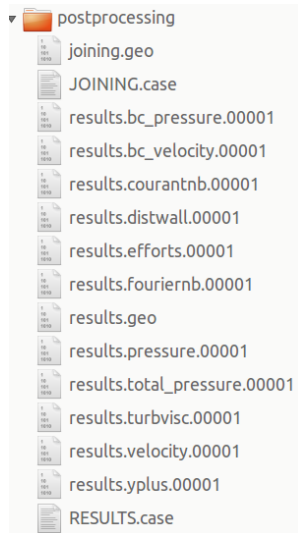
- A case file with info on files RESULTS.case.



Results: postprocessing files



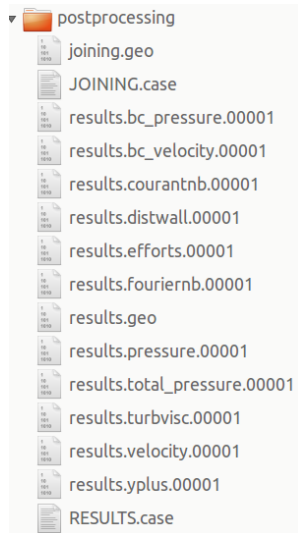
- A case file with info on files RESULTS.case.
- A geometry file results.geo.



Results: postprocessing files



- A case file with info on files RESULTS.case.
- A geometry file results.geo.
- One file per variable per time iteration (frequency set by the variable NTCHR in *cs_user_parameters.f90* or in the *Calculation Control - Output control* page of the GUI.



Results: postprocessing files



- A case file with info on files RESULTS.case.
- A geometry file results.geo.
- One file per variable per time iteration (frequency set by the variable NTCHR in `cs_user_parameters.f90` or in the *Calculation Control - Output control* page of the GUI).
- Files can be opened using ParaView (open-source under GPL).



- A case file with info on files RESULTS.case.
- A geometry file results.geo.
- One file per variable per time iteration (frequency set by the variable NTCHR in `cs_user_parameters.f90` or in the *Calculation Control - Output control* page of the GUI).
- Files can be opened using ParaView (open-source under GPL).
- *Parts* can be created by the user for specific sections of the geometry.

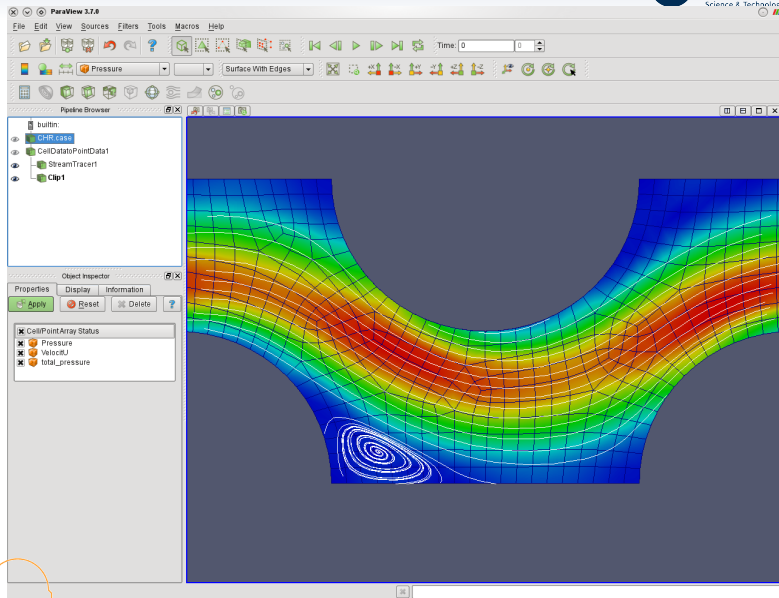


Results: CHR.ENSIGHT files



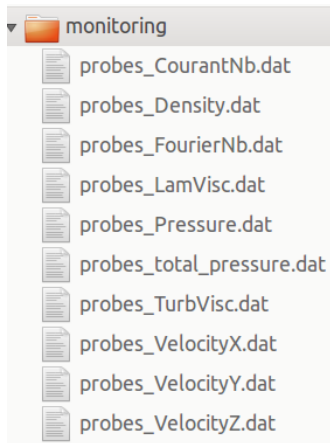
Hartree Centre

Science & Technology Facilities Council



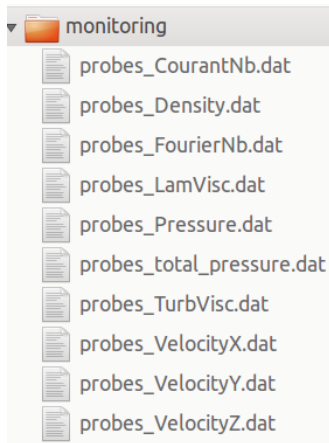


- One file per variable.



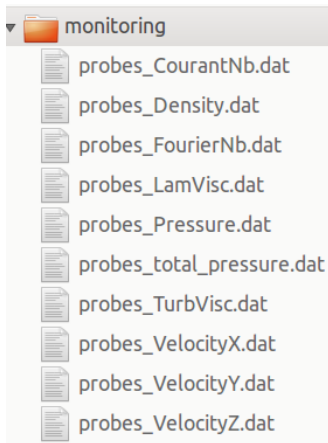


- One file per variable.
- Variation over number of iterations.

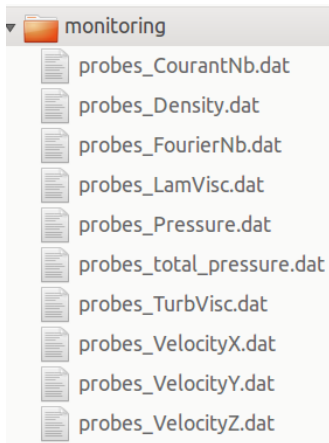




- One file per variable.
- Variation over number of iterations.
- Can be visualised via `xmgrace` (2D open source plot program).



- One file per variable.
- Variation over number of iterations.
- Can be visualised via `xmgrace` (2D open source plot program).
- Monitoring points defined in `usini1.f90` or in *Calculation control - Output control* page in the GUI.



Results: History files



Hartree Centre

Science & Technology Facilities Council

